HỌC VIỆN CÔNG NGHỆ BƯUCHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO

XÂY DỰNG CÁC HỆ THỐNG NHÚNG

ĐỀ TÀI: Hệ thống băng chuyền phân loại hàng hoá

Giảng viên:	**Đỗ Tiến Dũng**
Nhóm môn học:	**02**
Nhóm BTL:	**01**
Thành viên:	
Đỗ Bá Duy	B20DCCN148
Đỗ Tràng Lâm	B20DCCN388
Lê Mạnh Cường	B20DCCN100
Hoàng Minh Đức	B20DCCN196

Hà Nội - 2024

Mục lục

I. Giới thiệu chung:

Trong ngành công nghiệp hiện đại, hệ thống nhúng và băng chuyền phân loại mặt hàng tự động đóng vai trò quan trọng trong việc tối ưu hóa quy trình sản xuất và logistics. Với xu hướng ngày càng tăng cường tự động hóa, nhu cầu cho những hệ thống này ngày càng phát triển. Đòi hỏi từ các doanh nghiệp là đảm bảo độ chính xác và hiệu suất cao trong quá trình phân loại mặt hàng, cùng với tính linh hoạt để điều chỉnh cho các yêu cầu sản xuất đa dạng.

Sự tích hợp của trí tuệ nhân tạo và học máy không chỉ giúp tối ưu hóa quy trình phân loại, mà còn cải thiện khả năng dự đoán và đưa ra quyết định thông minh. Đồng thời, việc tuân thủ các tiêu chuẩn an toàn và môi trường là một yêu cầu không thể thiếu trong thiết kế và vận hành của các hệ thống này.

Thêm vào đó, sự tích hợp của Internet of Things (IoT) và quản lý dữ liệu thông minh giúp cải thiện khả năng giám sát và quản lý hiệu suất của hệ thống. Tất cả những yếu tố này cùng nhau tạo nên một môi trường sản xuất hiệu quả và linh hoạt, đáp ứng được nhu cầu đa dạng và nghiêm ngặt của thị trường ngày nay.

II. Lý do chọn đề tài

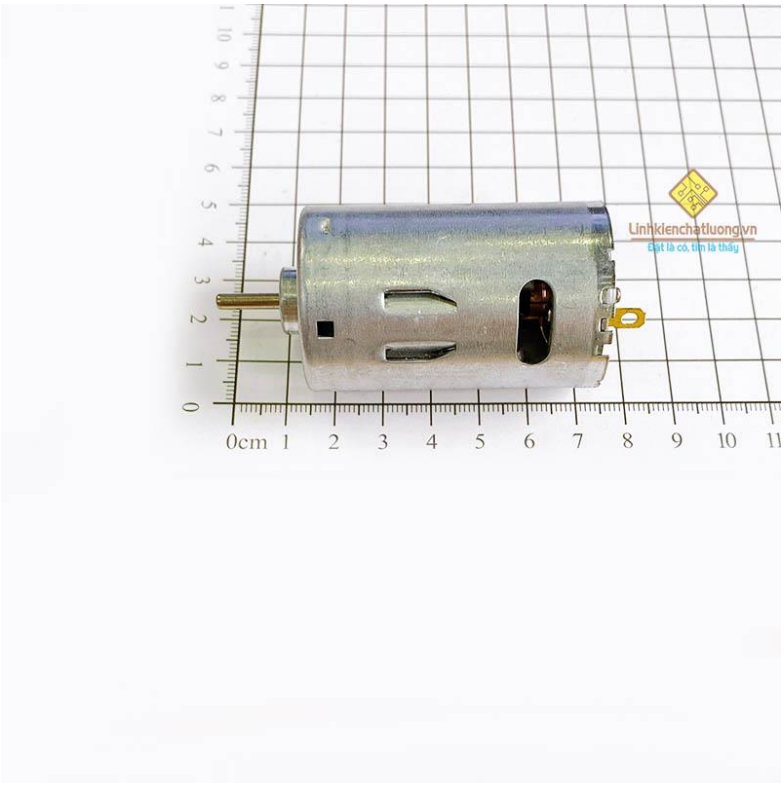
Trong thời đại công nghệ và kỹ thuật phát triển ngày nay thì nền công nghiệp thế giới nói chung hay ở Việt Nam nói riêng yêu cầu các quá trình tự động hoặc bán tự động. Ở các nhà máy hay các điểm bán hàng online thì việc vận chuyển hàng hoá không thể thiếu, để vận chuyển được thì sẽ phải cần hệ thống băng chuyền để vận chuyển hàng hoá và trí tuệ nhân tạo (AI) sẽ được tích hợp vào các hệ thống tự động để hỗ trợ trong việc phân loại hàng hoá khi được vận chuyển ở trên băng truyền.

III. Các thiết bị phần cứng:

1. Ống nhựa:

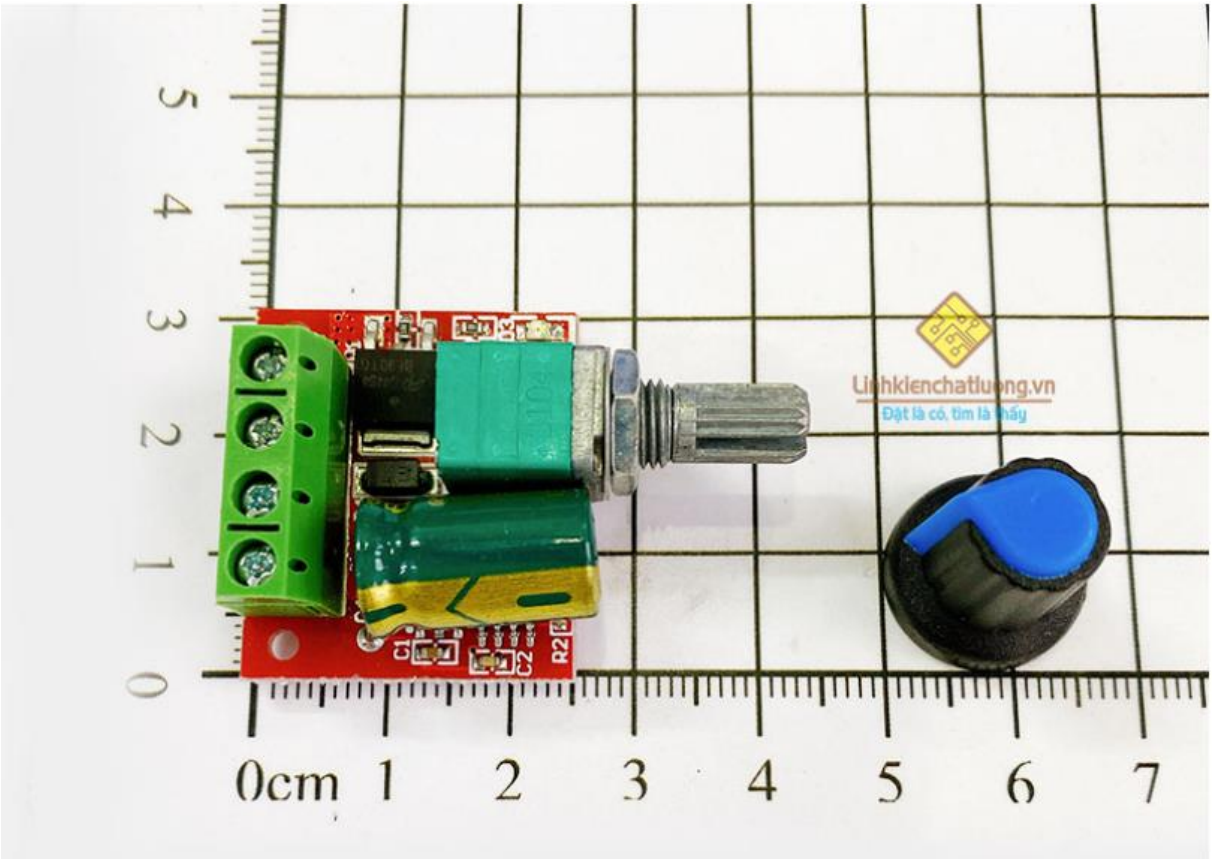
- Làm 2 bánh quay của băng chuyền

2. Động cơ motor bước quay chậm:



- Nguồn quay của băng chuyền

3. Bộ điều khiển động cơ bước:



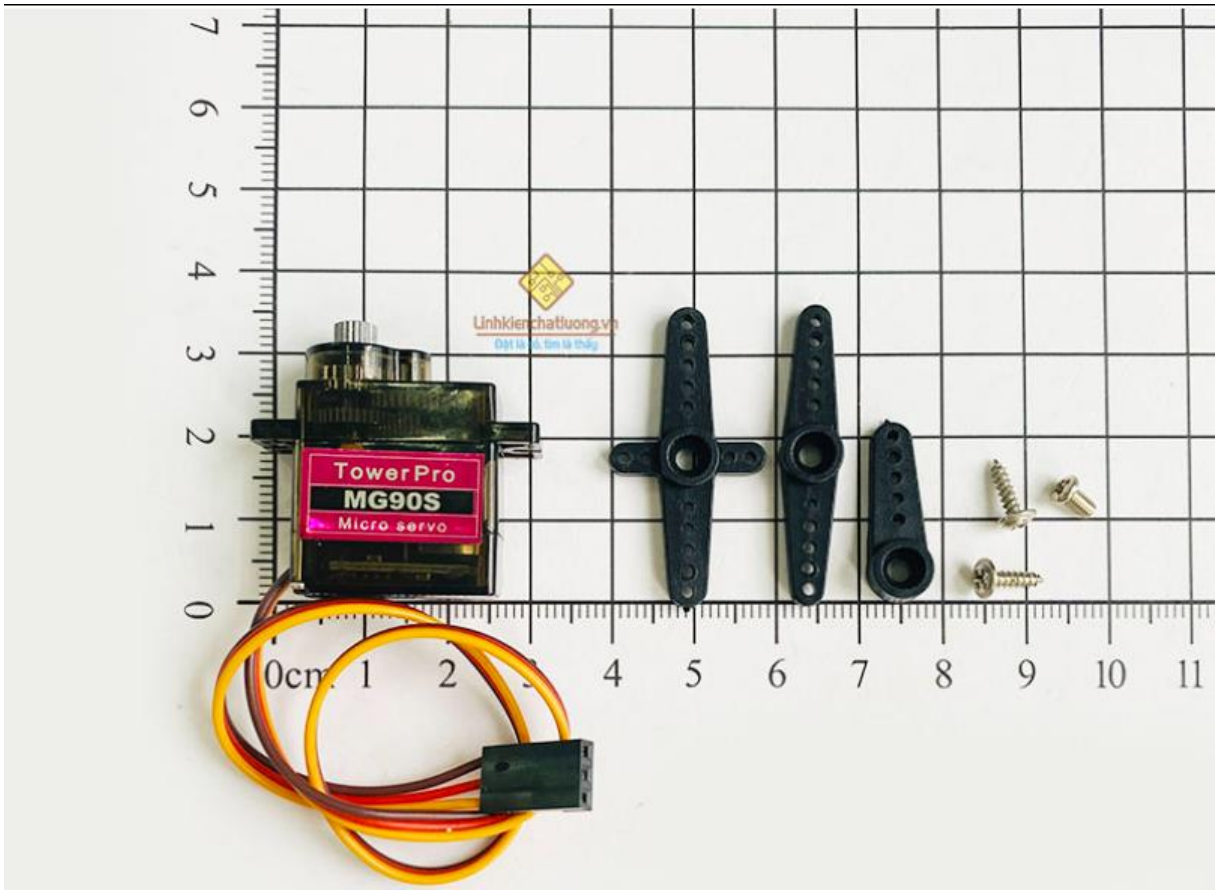
- Điều khiển tốc độ quay của động cơ bước.

- Giá đỡ lắp động cơ.

4. Vải da

- Dùng làm bề mặt của băng chuyền

5. Động cơ servo quay



- Rẽ hướng băng chuyền cho từng sản phẩm được phân loại

6. ESP32

- Khối điều khiển hệ thống

7. ESPCam

- Chịu trách nhiệm quét hình ảnh các sản phẩm cần phân loại.

IV. Luồng hoạt động của hệ thống:

Hệ thống băng chuyền phân loại sản phẩm tự động:

Các sản phẩm được đưa vào băng chuyền từ một cổng. Sản phẩm được đưa đi theo hướng băng chuyền di chuyển. Khi đó sản phẩm sẽ được quét qua cổng của ESP Cam. ESP cam truyền tải hình ảnh của sản phẩm đến hệ thống để phân tích hình ảnh và xử lý thông tin. Sau khi xử lí thông tin xong hệ thống gửi tín hiệu đến ESP32 điều khiển servo quay theo hướng cổng phân loại tương ứng của sản phẩm phân loại.

V. Code Arduino:

1. Kết nối ESP32 Cam với Servo.

```
#include <Servo.h>
#include <WiFi.h>
#include "esp_camera.h"

#define SERVO_PIN 2
#define CAM_PIN 4

Servo servo;

const char* ssid = "your_network_name";
const char* password = "your_network_password";

void setup() {
  Serial.begin(115200);

  // Khởi tạo servo
  servo.attach(SERVO_PIN);

  // Khởi tạo camera
  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = CAM_PIN;
  config.pin_d1 = -1;
```

```
config.pin_d2 = -1;
config.pin_d3 = -1;
config.pin_d4 = -1;
config.pin_d5 = -1;
config.pin_d6 = -1;
config.pin_d7 = -1;
config.pin_xclk = -1;
config.pin_pclk = -1;
config.pin_vsync = -1;
config.pin_href = -1;
config.pin_sscb_sda = -1;
config.pin_sscb_scl = -1;
config.pin_pwdn = -1;
config.pin_reset = -1;
config.xclk_freq_hz = 20000000;
config.pixel_format = PIXFORMAT_JPEG;

// Khởi tạo camera
if (esp_camera_init(&config) != ESP_OK) {
  Serial.println("Không thể khởi tạo camera");
  return;
}

// Kết nối WiFi
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
  delay(1000);
```

```
    Serial.println("Kết nối WiFi...");
  }
  Serial.println("Kết nối WiFi thành công");
}

void loop() {
  // Chụp ảnh từ camera
  camera_fb_t * fb = esp_camera_fb_get();
  if (!fb) {
    Serial.println("Không thể lấy hình ảnh từ camera");
    return;
  }

  // Xử lý hình ảnh ở đây để nhận diện sản phẩm
  // Xác định loại sản phẩm (giả sử đã xác định được loại sản phẩm)
  int productType = determineProductType();

  // Gửi tín hiệu điều khiển servo để phân loại sản phẩm vào cổng tương ứng
  controlServo(productType);

  // Giải phóng bộ nhớ hình ảnh
  esp_camera_fb_return(fb);

  delay(1000); // Thời gian chờ trước khi chụp ảnh tiếp theo
}

int determineProductType() {
```

```
  // Code để xác định loại sản phẩm dựa trên hình ảnh chụp được từ camera
  // Return product type (1, 2, 3, etc.)
  // Trong ví dụ này, ta chỉ làm một giả định
  return random(1, 4); // Trả về một giá trị ngẫu nhiên từ 1 đến 3
}
//Hàm điều khiển servo dựa trên loại sản phẩm
void controlServo(int productType) {
  // Điều chỉnh góc quay của servo để đưa sản phẩm vào cổng phân loại tương ứng (3 sản phẩm)
  int angle;
  switch (productType) {
    case 1:
      angle = 0; // Góc quay để đưa sản phẩm vào cổng phân loại 1
      break;
    case 2:
      angle = 90; // Góc quay để đưa sản phẩm vào cổng phân loại 2
      break;
    case 3:
      angle = 180; // Góc quay để đưa sản phẩm vào cổng phân loại 3
      break;
    // Thêm các trường hợp khác tùy thuộc vào số lượng cổng phân loại
    default:
      angle = 180; // Góc quay mặc định hoặc xử lý lỗi
      break;
  }
  servo.write(angle); // Điều chỉnh servo
  delay(1000); // Thời gian servo di chuyển và sản phẩm rơi vào cổng
```

}

2. Live Video Streaming

- Ta sẽ chạy các đoạn code dưới đây để lấy video streaming từ ESP32 cam và gửi lên trên server.

- Trước khi chạy code cần tải xuống tệp zip của projcet sau đó import thư viện vào trong library của Arduino.

https://github.com/yoursunny/esp32cam

- Board sử dụng trong Arduino là ESP32 Wrover Module.

2.1 Code gửi hình ảnh từ ESP32 Cam trong Arduino:

```
// Import thư viện
#include <WebServer.h>
#include <WiFi.h>
#include <esp32cam.h>

// Cấu hình wifi và thiết lập server.
const char* WIFI_SSID = "No Internet";
const char* WIFI_PASS = "nfpt1111";
WebServer server(80);

// Định nghĩa các hàm xử lý ảnh.
static auto loRes = esp32cam::Resolution::find(320, 240);
static auto midRes = esp32cam::Resolution::find(350, 530);
static auto hiRes = esp32cam::Resolution::find(800, 600);
void serveJpg()
{
  auto frame = esp32cam::capture();
  if (frame == nullptr) {
    Serial.println("CAPTURE FAIL");
```

```
    server.send(503, "", "");
    return;
  }
  Serial.printf("CAPTURE OK %dx%d %db\n", frame->getWidth(), frame->getHeight(),
                static_cast<int>(frame->size()));

  server.setContentLength(frame->size());
  server.send(200, "image/jpeg");
  WiFiClient client = server.client();
  frame->writeTo(client);
}

void handleJpgLo()
{
  if (!esp32cam::Camera.changeResolution(loRes)) {
    Serial.println("SET-LO-RES FAIL");
  }
  serveJpg();
}

void handleJpgHi()
{
  if (!esp32cam::Camera.changeResolution(hiRes)) {
    Serial.println("SET-HI-RES FAIL");
  }
  serveJpg();
}
```

```
void handleJpgMid()
{
  if (!esp32cam::Camera.changeResolution(midRes)) {
    Serial.println("SET-MID-RES FAIL");
  }
  serveJpg();
}

 // Cấu hình và khởi tạo camera.
void  setup(){
  Serial.begin(115200);
  Serial.println();
  {
    using namespace esp32cam;
    Config cfg;
    cfg.setPins(pins::AiThinker);
    cfg.setResolution(hiRes);
    cfg.setBufferCount(2);
    cfg.setJpeg(80);

    bool ok = Camera.begin(cfg);
    Serial.println(ok ? "CAMERA OK" : "CAMERA FAIL");
  }
  WiFi.persistent(false);
  WiFi.mode(WIFI_STA);
  WiFi.begin(WIFI_SSID, WIFI_PASS);
```

```
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
  }
  Serial.print("http://");
  Serial.println(WiFi.localIP());
  Serial.println("  /cam-lo.jpg");
  Serial.println("  /cam-hi.jpg");
  Serial.println("  /cam-mid.jpg");

// Set chất lượng của các hình ảnh (Low, High, Mid)
  server.on("/cam-lo.jpg", handleJpgLo);
  server.on("/cam-hi.jpg", handleJpgHi);
  server.on("/cam-mid.jpg", handleJpgMid);

  server.begin();
}

void loop()
{
  server.handleClient();
}
```

2.2 Đoạn code hiển thị luồng video từ link server:

Đoạn code này ta sẽ sử dụng python để chạy chương trình và ta cần cài các thư viện liên quan như openCV, numpy và thư viện url request.

```
import cv2
import urllib.request
import numpy as np
```

```
url = 'http://192.168.43.219/cam-hi.jpg' // Thay thế bằng URL được hiển thị trong Serial Monitor Arduino.
cv2.namedWindow("live Cam Testing", cv2.WINDOW_AUTOSIZE)



# Create a VideoCapture object
cap = cv2.VideoCapture(url)

# Check if the IP camera stream is opened successfully
if not cap.isOpened():
    print("Failed to open the IP camera stream")
    exit()

# Read and display video frames
while True:
    # Read a frame from the video stream
    img_resp=urllib.request.urlopen(url)
    imgnp=np.array(bytearray(img_resp.read()),dtype=np.uint8)
    #ret, frame = cap.read()
    im = cv2.imdecode(imgnp,-1)

    cv2.imshow('live Cam Testing',im)
    key=cv2.waitKey(5)
    if key==ord('q'):
        break
```

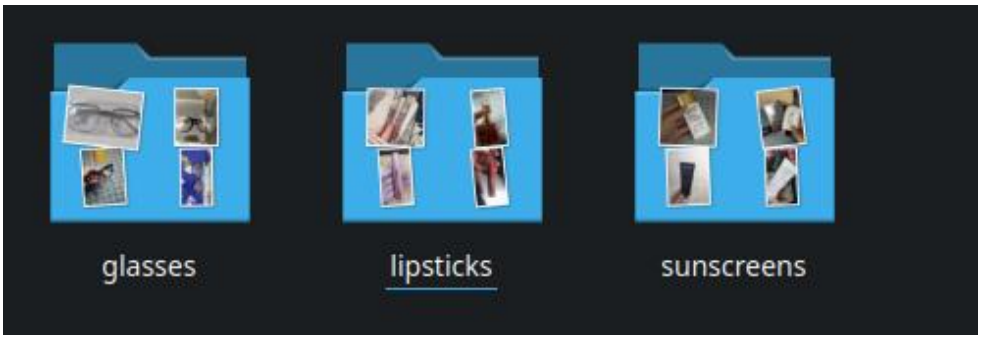
cap.release()

cv2.destroyAllWindows()

VI. Mô hình phân loại vật thể

1. Thu thập dữ liệu:

Số nhãn phân loại là 3 gồm kính mắt, son môi và kem chống nắng.



Hình ảnh dùng để train model sẽ được lấy từ các nguồn như google image và sàn thương mại điện tử Shopee. Sau khi lấy xong sẽ được phân loại để loại bỏ các ảnh không đạt tiêu chuẩn để train và sau khi loại bỏ các ảnh không đạt thì mỗi nhãn sẽ có khoảng 1000 mẫu. Sau đó các ảnh sẽ được được đánh dấu bounding box với Label Studio và gắn nhãn cho hình ảnh.

2. Chọn mô hình:

Ở trong hệ thống nhận diện ta sẽ sử dụng mô hình AI YoLoV5 để nhận diện vận thể.

Source code: https://github.com/ultralytics/yolov5

3. Chia dữ liệu thanh tập train và test

Dữ liệu sau khi được gán nhãn bằng tay xong sẽ được chia thành train và validation theo format của mô hình yolov5:

[26]:
```
import glob
import random
import os
import shutil

# Get all paths to your images files and text files
PATH = 'all data/dataset/'
img_paths = glob.glob(PATH + '*.jpg') + glob.glob(PATH + '*.jpeg') + glob.glob(PATH + '*.png')
txt_paths = glob.glob(PATH+'*.txt')
```

[27]:
```
print('img: ',len(img_paths),' txt: ',len(txt_paths))
```

```
img:  2849  txt:  2849
```

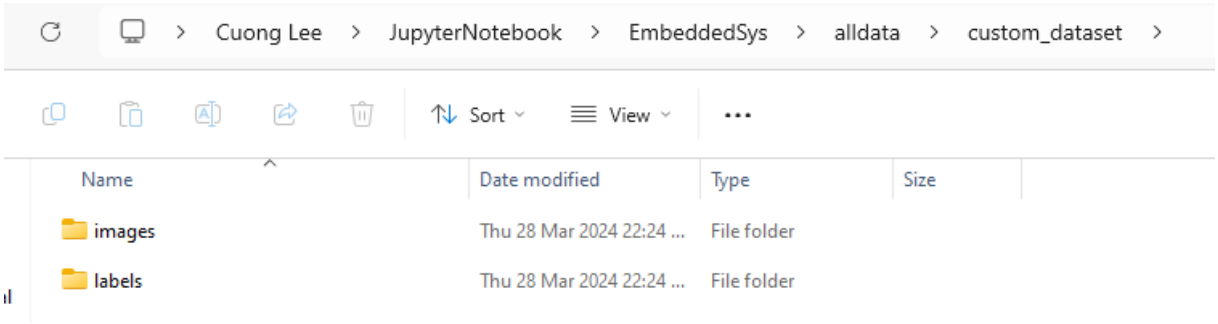
[28]:
```
# Calculate number of files for training, validation
data_size = len(img_paths)
r = 0.8
train_size = int(data_size * 0.8)
```

[29]:
```
# Shuffle two list
img_txt = list(zip(img_paths, txt_paths))
random.seed(13)
random.shuffle(img_txt)
img_paths, txt_paths = zip(*img_txt)
```

[30]:
```
# Now split them
train_img_paths = img_paths[:train_size]
train_txt_paths = txt_paths[:train_size]

valid_img_paths = img_paths[train_size:]
valid_txt_paths = txt_paths[train_size:]
```

[31]:
```
# Move them to train, valid folders
train_folder = PATH+'train/'
valid_folder = PATH+'valid/'
os.mkdir(train_folder)
os.mkdir(valid_folder)

def move(paths, folder):
    for p in paths:
        shutil.move(p, folder)

move(train_img_paths, train_folder)
move(train_txt_paths, train_folder)
move(valid_img_paths, valid_folder)
move(valid_txt_paths, valid_folder)
```

Sau khi chạy đoạn lệnh:



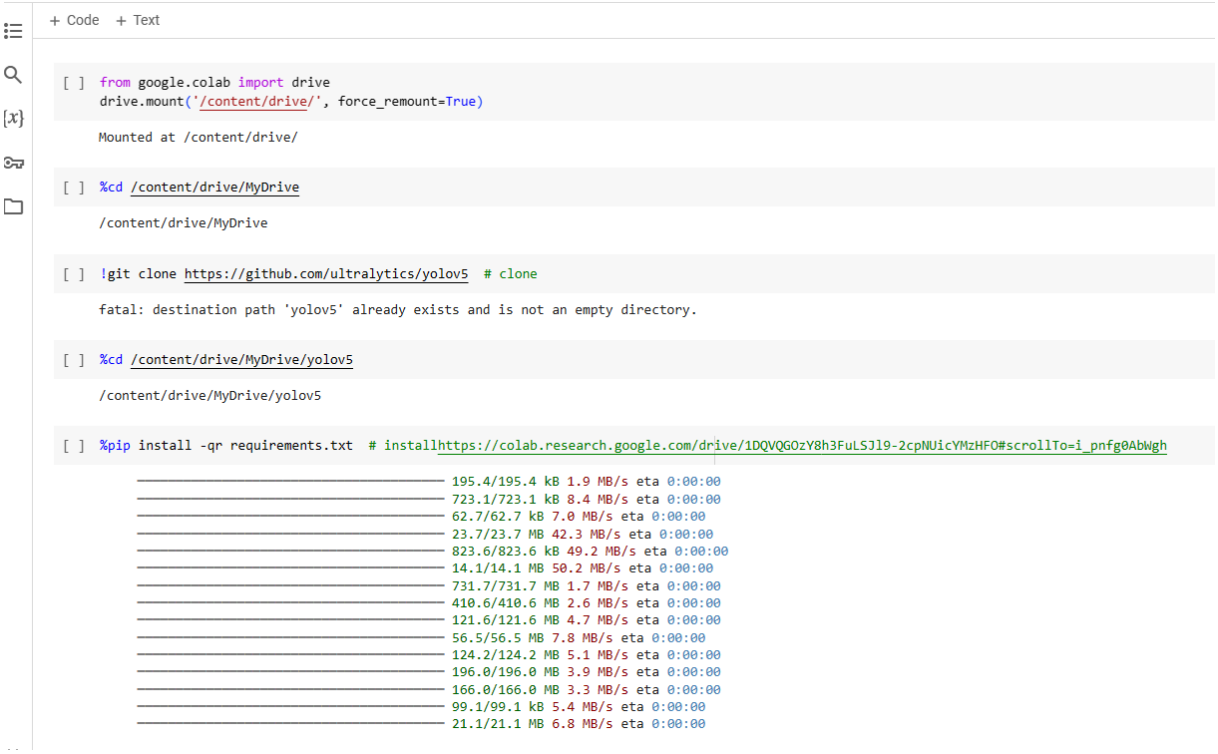
4. Tiền xử lý dữ liệu

Do dữ liệu được gán nhãn bằng tay cho nên ta sẽ chỉ cần upload lên colab để train, ở đây sử dụng google colab do colab cho phép "mượn" GPU của colab để train. Một phần cũng do GPU colab cũng tương đối mạnh, mạnh hơn laptop thông thường.

5. Huấn luyện mô hình

Sử dụng drive để lưu mô hình sau khi train nên cần mount dữ liệu

Tiếp đến ta cài đặt các thư viện yêu cầu của mô hình:



Kiểm tra GPU và giải nén các ảnh:

```
from drive.MyDrive.yolov5 import utils
display = utils.notebook_init()  # checks
```

```
YOLOv5 🚀 v7.0-295-gac6c4383 Python-3.10.12 torch-2.2.1+cu121 CUDA:0 (Tesla T4, 15102MiB)
Setup complete ✅ (2 CPUs, 12.7 GB RAM, 28.9/78.2 GB disk)
```

```
[ ] !nvidia-smi
```

```
/bin/bash: line 1: nvidia-smi: command not found
```

```
[ ] !unrar x "/content/drive/MyDrive/custom_dataset.rar" "./data"
```

```
Extracting  ./data/custom_dataset/images/train/513fbcc7-gong-kinh-povino-po1377-2-300x300.jpg  OK
Extracting  ./data/custom_dataset/images/train/51600d6a-images52.jpg  OK
Extracting  ./data/custom_dataset/images/train/5179af14-2e4c39ae0d0a8ab80ad51fef0d1d0c86.jpg  OK
Extracting  ./data/custom_dataset/images/train/51858858-SunScreen_122.jpg  OK
Extracting  ./data/custom_dataset/images/train/519d93cc-image26.jpeg  OK
Extracting  ./data/custom_dataset/images/train/51c407b1-SunScreen_63.jpg  OK
Extracting  ./data/custom_dataset/images/train/51ca6c3f-0123546dca48f0dc94c6ff9242fd14eb.jpg  OK
Extracting  ./data/custom_dataset/images/train/5208d001-Glasses_297.jpg  OK
Extracting  ./data/custom_dataset/images/train/52193a31-LipSticks_250.jpg  OK
Extracting  ./data/custom_dataset/images/train/521f94bf-images78.jpg  OK
Extracting  ./data/custom_dataset/images/train/5223373e-GONG-FURLA-VFU581.png  OK
Extracting  ./data/custom_dataset/images/train/5259a20a-SunScreen_142.jpg  OK
Extracting  ./data/custom_dataset/images/train/526ba3e7-cc960d876664faaa5032e4e576c20511.jpg  OK
Extracting  ./data/custom_dataset/images/train/527c049b-Glasses_274.jpg  OK
Extracting  ./data/custom_dataset/images/train/52b35bab-Glasses_119.jpg  OK
Extracting  ./data/custom_dataset/images/train/52c2a36b-images54.jpg  OK
Extracting  ./data/custom_dataset/images/train/52dbfcd8-images53.jpg  OK
```

Bắt đầu quá trình huấn luyện với ảnh sẽ được resize thành 640x640, một mẻ 16 ảnh, 150 vòng, theo 3 nhãn được định nghĩa ban đầu của file .yaml và sử dụng pretrained model yolov5s.pt

```
# Train YOLOv5
!python train.py --img 640 --batch 16 --epochs 150 --data custom_data.yaml --weights yolov5s.pt
```

```
2024-03-28 16:33:45.910358: E external/local_xla/xla/stream_executor/cuda/cuda_dnn.cc:9261] Unab
2024-03-28 16:33:45.910438: E external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:607] Unabl
2024-03-28 16:33:45.912421: E external/local_xla/xla/stream_executor/cuda/cuda_blas.cc:1515] Una
train: weights=yolov5s.pt, cfg=, data=custom_data.yaml, hyp=data/hyps/hyp.scratch-low.yaml, epoc
github: up to date with https://github.com/ultralytics/yolov5 ✅
YOLOv5 🚀 v7.0-295-gac6c4383 Python-3.10.12 torch-2.2.1+cu121 CUDA:0 (Tesla T4, 15102MiB)

hyperparameters: lr0=0.01, lrf=0.01, momentum=0.937, weight_decay=0.0005, warmup_epochs=3.0, war
Comet: run 'pip install comet_ml' to automatically track and visualize YOLOv5 🚀 runs in Comet
TensorBoard: Start with 'tensorboard --logdir runs/train', view at http://localhost:6006/
Overriding model.yaml nc=80 with nc=3

                 from  n    params  module                                  arguments
  0                -1  1      3520  models.common.Conv                      [3, 32, 6, 2, 2]
  1                -1  1     18560  models.common.Conv                      [32, 64, 3, 2]
```

6. Đánh giá và điều chỉnh mô hình

```
libpng warning: iCCP: known incorrect sRGB profile
     24/149      4.69G    0.03589    0.01742   0.007014         23        640:  84% 120/143 [01:41<00:16,  1.43it/s]libpng warning: iCCP: known incorrect sRGB profile
     24/149      4.69G    0.03588    0.01734   0.007563          7        640: 100% 143/143 [02:02<00:00,  1.17it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95:  17% 3/18 [00:01<00:08,  1.84it/s]libpng warning: iCCP: known incorrect sRGB profile
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 18/18 [00:11<00:00,  1.57it/s]
                   all        570        173       0.17      0.668      0.203      0.127

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
     25/149      4.69G     0.0359    0.01653   0.005728         48        640:  36% 52/143 [00:43<01:24,  1.08it/s]libpng warning: iCCP: known incorrect sRGB profile
     25/149      4.69G    0.03548    0.01693   0.005599         39        640:  48% 68/143 [00:56<01:08,  1.09it/s]libpng warning: iCCP: known incorrect sRGB profile
     25/149      4.69G    0.03495    0.01701    0.00519         36        640:  57% 82/143 [01:08<00:48,  1.27it/s]libpng warning: iCCP: known incorrect sRGB profile
     25/149      4.69G    0.03511    0.01721   0.005158         45        640:  62% 88/143 [01:13<00:37,  1.47it/s]libpng warning: iCCP: known incorrect sRGB profile
     25/149      4.69G    0.03506    0.01722    0.00514         38        640:  63% 90/143 [01:15<00:37,  1.41it/s]libpng warning: iCCP: known incorrect sRGB profile
     25/149      4.69G     0.0354    0.01718   0.004664          7        640: 100% 143/143 [01:59<00:00,  1.19it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95:  11% 2/18 [00:01<00:09,  1.66it/s]libpng warning: iCCP: known incorrect sRGB profile
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 18/18 [00:12<00:00,  1.49it/s]
                   all        570        173      0.187      0.738      0.213      0.142

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
     26/149      4.69G    0.03699    0.01662   0.004121         42        640:  31% 45/143 [00:36<01:13,  1.34it/s]libpng warning: iCCP: known incorrect sRGB profile
     26/149      4.69G    0.03666    0.01722   0.004217         39        640:  48% 69/143 [00:56<00:49,  1.50it/s]libpng warning: iCCP: known incorrect sRGB profile
     26/149      4.69G    0.03644    0.01715    0.00421         34        640:  49% 70/143 [00:58<01:10,  1.04it/s]libpng warning: iCCP: known incorrect sRGB profile
     26/149      4.69G    0.03662    0.01718   0.004142         32        640:  59% 85/143 [01:10<00:42,  1.36it/s]libpng warning: iCCP: known incorrect sRGB profile
     26/149      4.69G    0.03652    0.01716   0.004148         42        640:  68% 97/143 [01:21<00:32,  1.41it/s]libpng warning: iCCP: known incorrect sRGB profile
     26/149      4.69G    0.03628    0.01703   0.004117         44        640:  73% 104/143 [01:28<00:45,  1.16s/it]libpng warning: iCCP: known incorrect sRGB profile
     26/149      4.69G    0.03586    0.01698   0.004072         54        640:  79% 113/143 [01:35<00:21,  1.37it/s]libpng warning: iCCP: known incorrect sRGB profile
     26/149      4.69G    0.03573    0.01697   0.004118         46        640:  80% 115/143 [01:36<00:19,  1.47it/s]libpng warning: iCCP: known incorrect sRGB profile
     26/149      4.69G    0.03576    0.01691   0.004004         46        640:  92% 131/143 [01:50<00:08,  1.40it/s]libpng warning: iCCP: known incorrect sRGB profile
     26/149      4.69G    0.03553    0.01687   0.003928         26        640:  96% 137/143 [01:56<00:05,  1.12it/s]libpng warning: iCCP: known incorrect sRGB profile
     26/149      4.69G     0.0354    0.01687   0.003923         12        640: 100% 143/143 [02:01<00:00,  1.18it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95:  11% 2/18 [00:01<00:09,  1.75it/s]libpng warning: iCCP: known incorrect sRGB profile
                 Class     Images  Instances          P          R      mAP50   mAP50-95:  22% 4/18 [00:02<00:07,  1.83it/s]libpng warning: iCCP: known incorrect sRGB profile
                 Class     Images  Instances          P          R      mAP50   mAP50-95:  61% 11/18 [00:06<00:03,  1.94it/s]
```

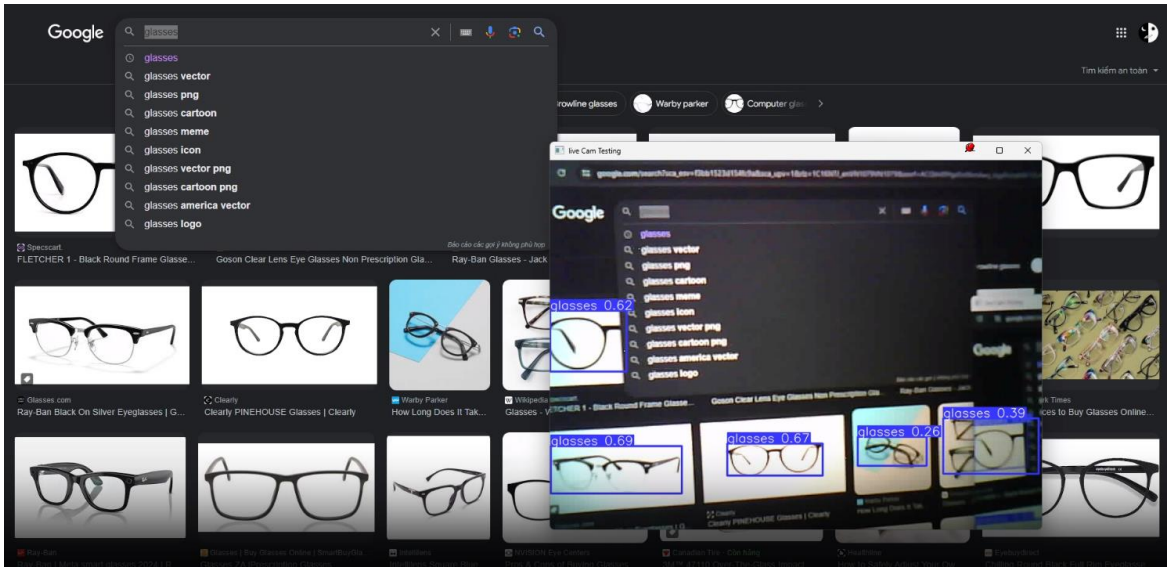
7. Kiểm tra mô hình

8. Triển khai mô hình

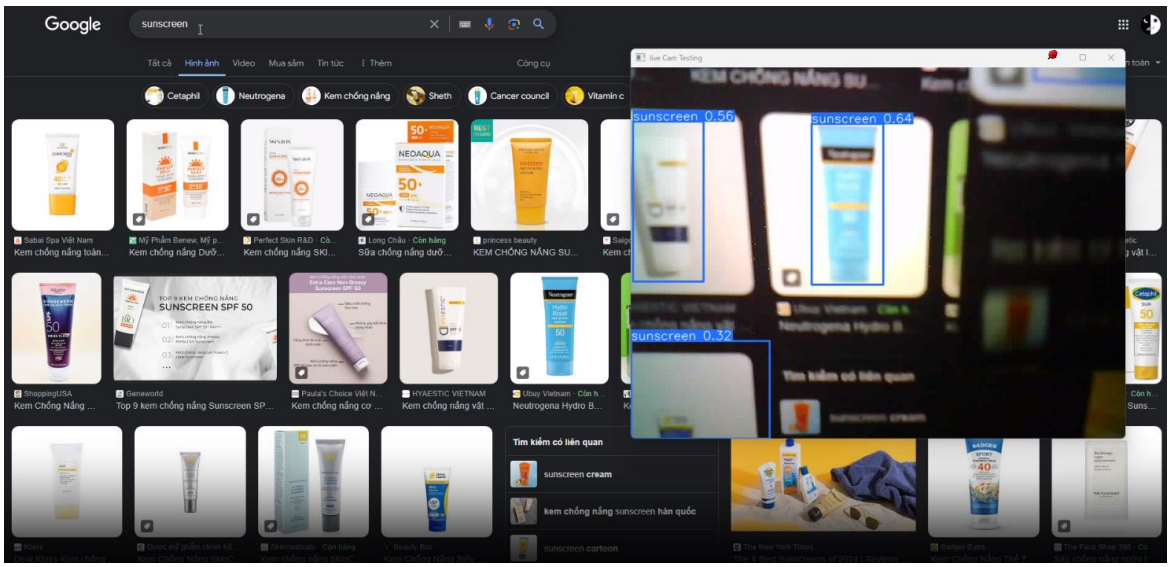
VII. Kết hợp mô hình AI với việc sử dụng ESP32 Cam.

VIII. Kết quả tạm thời thu được

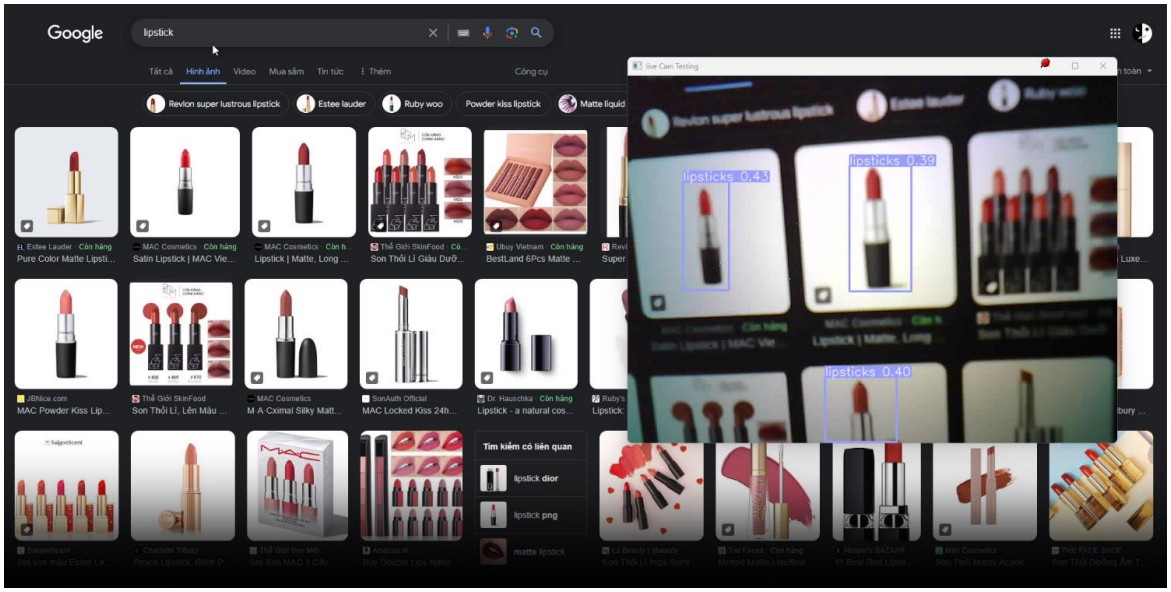
Hiện tại hệ thống đã có thể lấy video từ luồng hình ảnh từ ESP32 Cam sau đó sử dụng mô hình AI để nhận diện ra 3 nhãn vật thể được phân loại.



Hình 7.1: Nhận diện kính mắt từ hình ảnh ESP32 Cam.



Hình 7.2: Nhận diện kem chống nắng từ hình ảnh ESP32 Cam.



Hình 7.3: Nhận diện son môi từ hình ảnh ESP32 Cam.